Test Methodology for the McKinley Processor

Don Douglas Josephson, Hewlett-Packard Company
Steve Poehlman, Intel Corporation
Vincent Govan, Hewlett-Packard Company
Clint Mumford, Hewlett-Packard Company

Introduction

The McKinley processor is the result of a joint design effort between Intel and Hewlett-Packard engineers, and is the second processor implementation of the Itanium™ Processor Family (IPF) architecture. This paper describes the methodology developed for testing a complex high-performance microprocessor design. An overview of the processor is presented, along with the goals for the test methodology. Details of the test control blocks, scan methodology, and clocking are given. The scanlatch design, tradeoffs and verification processes are discussed, along with some details of ATPG modeling and memory array testing. Finally, some results are presented.

Processor Overview

The McKinley processor is fabricated in Intel's 0.18 µm process [1], and consists of over 220 million transistors. The processor implements the IPF architecture jointly developed by Intel and HP [2], with a brand new microarchitectural implementation designed to achieve industry-leading performance [3]. Using a short 8-stage pipeline and 3 levels of cache hierarchy, including a 3 MB on-chip L3 cache, a 256 kB unified L2 cache and 2 16 kB L1I and L1D single-cycle caches, the processor is designed to operate at frequencies in the range of 1 GHz. Up to 6 instructions can be executed at once, and over 20 operations of various types can be performed per clock cycle. Off-chip communication is accomplished via a 200 MHz bus, which includes a double-pumped 400 MT/s 128-bit data bus.

The processor offers binary backward compatibility with the Intel IA-32™ architecture and is capable of running IA-32 binaries directly, without modification. Seamless compatibility with HP PA-RISC™ binaries is achieved through software translation. Approximately 75% of the transistors in the design comprise RAM and CAM cells in 13 different arrays, which are designed to maximize performance and data bandwidth. The remaining transistors implement the core engine of the IPF architecture, along with the IA-32 compatibility engine.

Test Methodology Goals

There were five key goals driving the test methodology decisions for the McKinley processor:

- Reduce the design impact of adding test support circuitry
- Enable fast, efficient analysis of functional and electrical bugs in post-silicon debug
- Reduce the engineering time spent on generating manufacturing tests
- Increase the quality and coverage of manufacturing test
- Keep the test methodology simple and consistent

These goals were influenced by the increasingly large impacts that previous HP and Intel test methodologies [4-7] had placed on the design team in terms of time required for implementation, and the large amount of engineering resources required for test generation and post-silicon debug, which have been on an exponentially increasing curve over the past decade. These methodologies had been adequate for past Intel and HP processor designs, but it was clear that we needed to adopt new techniques if we were to be successful in achieving the goals described above.

Test Methodology Overview

Testing a complex microprocessor is a huge challenge [8-14]. Accordingly, we greatly increased the amount of on-chip support for testing and debugging of the processor as compared to previous designs at both Intel and HP [4-7]. Due to the wide variety of circuit design styles and use of new and aggressive circuit design techniques, we implemented nearly full scan in the design to support high quality and efficient manufacturing test as well as excellent observability for debug purposes. Also, due to the large amount of various kinds of memory structures on the chip, direct access to memory arrays over roughly 4 kB in size in the design (13 in all) was implemented via the off-chip bus interface. This allows excellent manufacturing test coverage for the arrays, as well as flexibility to design new test algorithms if new circuit

failure mechanisms are discovered. This capability can also be used for burn-in through built-in toggle and BIST features and is also accessible through the test controller. This capability reused access paths already needed for diagnostics and initialization of the arrays in the microarchitecture.

Integrated Test Controller

The Integrated Test Controller (ITC) is the main test control block on the processor. The main purpose of the ITC is to implement IEEE Standard 1149.1 access to the device as well as to support the many custom test and debug features of the design. The ITC uses an 8-bit instruction register and 84 instructions to control test operations. It is controlled by the standard 1149.1 interface pins, consisting of test clock (TCK), test data in (TDI), test mode select (TMS), test data out (TDO), and test reset (TRST) pins. The McKinley processor complies with the IEEE 1149.1 boundary scan specification [15]. The public commands supported are SAMPLE/PRELOAD, EXTEST, BYPASS, and HIGHZ.

The ITC interfaces with distributed test controllers called "minitaps" spread throughout the chip via a dedicated test bus [16]. The minitaps receive TCK from a balanced H-tree distribution, and generate the shift signal for scanpaths that are active during test operations. They also provide local electrical buffering of test signals and route scan data from either the ITC or the bus interface of the chip depending on the scan access method used. A signal is generated to indicate when the scanpaths are shifting, which allows circuitry that may drive fight during shifting to be turned off. An I_{DDQ} enable signal is provided that allows circuits that draw static current to be shut down. While this capability was used, I_{DDQ} testing is not fully supported in the design, largely due to the large transistor count and the concomitant leakage (several amps).

The IEEE 1149.1 standard method of shifting scanpaths out of a chip is via TDI and TDO; however, only one scanpath can be accessed at a time via this mechanism. The processor also supports parallel access to many of the internal scanpaths via the bus interface. The ITC takes control of the bus during manufacturing test operations and shifts the scanpaths through these pins. This enables parallelization of production scan tests and reduces the amount of testing time required to test the device.

Scan Methodology

The scan methodology for the processor was chosen to support the overall goals of the test methodology while reducing the impact to the design as much as possible. Many diverse and challenging high performance circuit design techniques needed to be supported by the methodology to ensure adequate manufacturing testability as well as ease of silicon debug. A wide range of custom circuit techniques were used in the design of the processor, including static, pseudo and clocked NMOS, pulse clocked, single and dual-rail dynamic, self-timed and self-resetting circuits [17]. Integrating a consistent and easy to use scan methodology with this wide variety of circuit design techniques was a difficult challenge.

Support for scan for both static and dynamic circuits was implemented in the methodology. The design contains 51 scanpaths. 36 scanpaths have direct parallel access as previously described. Approximately 136,000 state elements are accessed via these scanpaths, which are also independently addressable via the ITC through the 1149.1 interface; the longest scanpath is 4800 bits long. It is difficult to precisely count the total number of "state elements" in the design, but an estimate of the total state element count for the processor would be roughly 160,000; therefore, approximately 85% of the internal state elements (exclusive of memory arrays and register files) are scan accessible. Many of these non-scanned elements are queues or are present for conversion between dynamic/static circuit domains. Five scanpaths accessing 24,000 non-destructive observe-only scanlatches allow access to internal state while the chip continues to operate. The longest of these scanpaths is 8000 bits long. These are used for special observability purposes during silicon debug [4 - pg. 771, 6 - pg. 298]. The rest of the scanpaths are accessed through the ITC, and access various other test features within the design.

Clocking and Latching

To understand the implementation of the scan methodology for the processor, some basic principles of the clocking and latching methodology of the processor must be described. In general, the design uses edge-triggered latches using pulse clock technology [18], where short pulses are used to clock transparent latches, effectively making them act as edge triggered flip-flops but without the additional power, delay and area penalties involved.

For dynamic circuitry, a novel latching capability to provide low-penalty transparent latching in the final gate of each dynamic logic phase was implemented by separately controlling the precharge and evaluate clocks. This also allowed integration of scan capability into the dynamic circuits without the traditional delay and area overhead of isolated transparent latches.

To clock these state elements, the processor distributes a single clock throughout the design via a low skew wire network across the device. The source clock (SLCBO) is driven from an on-chip phase locked loop (PLL) through several levels of buffering and routing into local regions. Within each region are many small clusters of individual "gaters" clocked by SLCBO which comprise the final level of clock buffering before state elements in the design. Gaters can also receive a qualifying signal to "gate" the local clocks driven to a set of latches (thus the name).

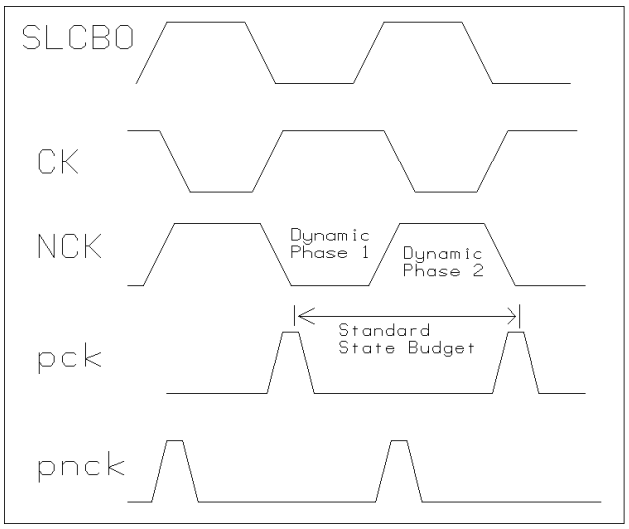

Figure 1 – McKinley Clock Waveforms

A variety of gaters were used in the design to generate different versions of local clocks (Figure 1) depending on circuit design requirements, but the basic classes of gater output clocks were either phase clocks which were typically used for phase clocked dynamic circuits (CK or NCK – logically opposite from each other), or pulse clocks which were used for static logic (pck or pnck – short pulses generated on the falling or rising edge of SLCBO respectively). The NCK clock is logically the same as the SLCBO clock delivered to the gaters.

Scanlatch Design

Static and dynamic latches are supported for scan operations (Figures 2, 3). During normal operation, static latches function as normal pulse latches. When the pck signal pulses, data propagates from the signal in into the latch circuit and onto the storage node q. The data is captured when pck goes away.

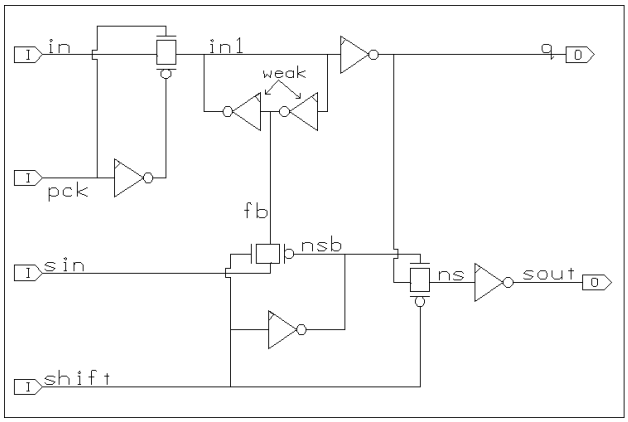

Figure 2 – Static Scanlatch Implementation

For dynamic circuits, the last dynamic gate has a dynamic latch converter (DLC) added to it to separately control the precharge (rck) and evaluate (eck) clocks and to add full keeper FETs to hold the value in the gate during the next phase of evaluation. The DLC also adds scan capability to the dynamic gate.

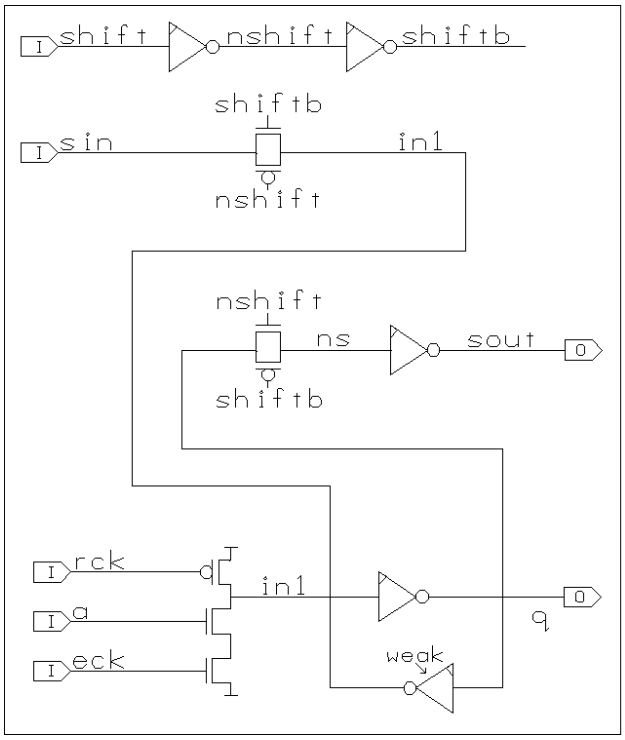

Figure 3 – Dynamic Scanlatch Implementation

During test operations, SLCBO is always halted low prior to the beginning of scanpath operations. The ITC can control SLCBO, halting or stepping it one or more times, allowing either single (Figure 4) or multiple-step testing to be done. This enables traditional stuck-at fault testing

with single clocks, as well as delay fault testing with multiple clocks.

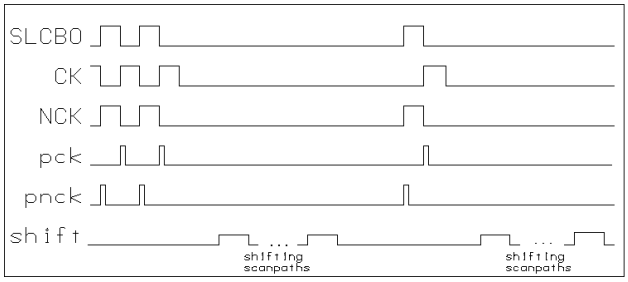

Figure 4 – Single Step Testing Operation

Since pck is always off due to its pulsed nature, for static latches it does not matter if SLCBO is halted high or low; the latch will always be isolated, enabling scanning of the latch while the clock is off. For DLCs, the rck and eck clocks are set to 1 and 0 respectively, tristating the precharge and evaluate paths and allowing scan to occur through the DLC. Only CK phase DLCs are scanned; since NCK is low, NCK dynamic gates downstream from CK DLCs cannot be affected by shifting operations as long as an NCK-controlled "footer" FET is always used for the first dynamic gate in the NCK phase of evaluation, which prevents the gate from evaluating as data is scanned through the previous CK DLC.

For both static and dynamic scanlatches during testing, shift is clocked to shift test data, as shown in Figure 4. Once the new data has been shifted in, all the latches are holding the desired values, and shift is low. The ITC then causes SLCBO to step as desired, which propagates the data through the logic between the latches and DLCs. Incoming data is captured into the latches/DLCs when pck pulses. Captured data is then scanned out and the process is repeated.

Scanlatch Advantages and Disadvantages

The primary advantage of this scanlatch design is the minimal amount of additional hardware (area/wires) necessary to make the latch scannable. shift, sin and sout are the only additional ports required in the cell. Typically, sin and sout can be connected by abutment to adjacent scanlatches, and shift can be routed over the entire stack of scanlatches, considerably reducing the area impact of scan. This is especially effective in datapaths with their regular layouts; the cost can be as low as one metal track for the shift line in higher metal, with sin and sout below it in metal 1 or diffusion. This kind of technique can also be used in control blocks, where latches can be preplaced in "stacks" prior to routing. However, even if scanlatches are randomly placed, the overhead is still quite low. shift is the only signal which is remotely timing critical, and it is only necessary to provide an adequately fast edge rate for it to avoid an internal race inside the scanlatch.

Fewer signals (especially timing critical signals) means less impact to the circuit designer, since things are easier to check and there are not as many issues to worry about. The extra circuitry required by the scanlatch is also much less than that required by previous HP and Intel methodologies. Only 8 additional small FETs are required for the latch design to be fully scannable, compared to 25-30 needed in a previous methodology [4 – pg. 767]. For the typical latch used in the design, this resulted in only a 20% area overhead per latch, which is considerably less than in previous designs at HP and Intel. As a result, the overall area impact for scanlatch overhead is less than 0.5% of the entire core circuit area of the processor, and had no impact on critical timing paths as a result. Also, due to the pulse nature of the pck signal, there is no requirement to gate the clock for test operations, which can be a performance impact. The speed impact of adding scan in the typical latch is less than 3 ps on flowthrough time. Thus, performance impact of nearly full scan is negligible (when properly designed), and does not lead to any significant area penalty.

One disadvantage of this scanlatch is the utilization of the functional latch as the master latch. This causes the output to "wiggle" during shift operations. Since the data flowing through the scanpath can be anything, q can transition from 0->1 or 1->0 at any time during the shift cycle. This creates a problem for pass gate multiplexer structures. Multiplexer fights can occur if multiple select lines are on at the same time. If there are a large number of these on simultaneously, with opposing data on the mux inputs, there can be a power problem during shifting due to contention. For small muxes, the mux artwork was robust enough to handle these kinds of fights. Larger muxes were identified with a tool and special scanlatches were used for these cases to only allow the scanlatch to update when shifting was complete, and ATPG is constrained to generate only "safe" patterns for such cases. Less than 0.3% of the scanlatches required this functionality. These contentions do not occur in normal operation.

There are three issues with the scanlatch circuit configuration. One is that leakage from the ns node could cause the inverter to change state (since there is no active feedback in order to save area), thus corrupting the captured value from q during shifting. This is easily managed by controlling the width of the shift pulse via a leakage monitoring circuit to allow the pulse width to

scale with leakage. This guarantees the shift signal will always fall prior to the ns node voltage degrading.

The second issue is that since shift is low during normal operation, when the output of the latch changes, the inverter connected to the ns node will also change state, and will charge/discharge any capacitance on the sout node. This results in additional spurious power dissipation in normal operation when latches change state. Fortunately in most cases this capacitance was negligible due to abutment of scanlatches. In cases where it was not, latches were used to isolate the sout capacitance and reduce the additional power impact. The contribution to normal power dissipation due to this is estimated to be considerably less than 0.5W at the nominal operating point, which was acceptable for the area savings realized in the scanlatch design.

The final issue is that there could be an internal race if q can change due to shift rising before the transmission gate connected to ns shuts off completely, or a race between adjacent scanlatches if shift on a subsequent latch cannot fall far enough in advance of nshift of the previous latch rising. These two conditions were rigorously checked in verification of the design.

Scan Circuitry Verification

Using a template-based SPICE evaluation tool across process, temperature, and voltage, each unique scanlatch design was verified to properly set and hold values through the serial scan inputs and outputs; each was also verified to be able to electrically scan data into any possible neighbor scanlatch. This tool was also used to check many other electrical performance parameters of each cell to ensure robust electrical operation of the cell for all parameters, not just scan. Use of this tool allowed designers freedom to create different types of latch cells for special purposes (e.g. FET sizing for delay, additional inputs to a latch) while still maintaining rigorous "best practice" design specifications. Several hundred unique scanlatch types were used in the design, but they were all simple variants of only a handful of basic scanlatch designs. Use of this SPICE-based tool allowed fast and easy verification of all "logically identical but electrically different" scanlatches and ensured excellent electrical robustness when shifting scanpaths in the design. Run time was tens of seconds for this tool for each scanlatch type it was run on.

Netlist stitching tools were used to connect schematic scanpaths as they were designed. This allowed ATPG to begin early in the design process to aid in verifying testability. Provisions were also made in the tool to allow back-annotation of artwork scan connectivity in synthesis blocks as well to enhance designer productivity.

A switch-level simulation tool was used to verify that the scanpaths were continuous and properly connected at any level of hierarchy in the design, from sub-block to chip-wide. This tool also verified that the final artwork scanpath connectivity was continuous and properly shifted data. Run time for this tool was anywhere from seconds to several hours depending on if it was run at a low level or at the chip level.

Finally, designers used a tool that accurately extracted and simulated the distribution of the scanpath shift clocks to all scan latches using SPICE. This was needed due to the sensitivity of the single shift signal of the scan latches to edge rate. The tool also verified that data did not race between adjacent scan latches, and that minimum frequency requirements were met. Run time for this tool at the highest level it was required to be run (up to several thousand scanlatches) was tens of minutes. These tools were simple to use and were successful in minimizing designer productivity impacts due to test, as well as ensuring that the scan circuitry was very robust.

ATPG Modeling

As the design progressed, ATPG was run on units as they became available in order to identify any possible testing difficulties. Circuits were remodeled using a mix of manual and automated techniques. Verilog models were created from physical design netlists. A library of cells was developed incrementally as more units became available and went through the process. Static pulse latches were modeled using pulse primitives in the ATPG tool and standard transparent latch primitives.

Remodeling synthesis blocks and generating good vectors (greater than 90% stuck-at fault coverage) was easy, as expected, but difficulties were encountered in custom datapaths, especially where dynamic logic was heavily used. Dynamic circuits were remodeled to combinational equivalents to simplify the job of remodeling and to make it easier for ATPG to "understand" how the circuitry worked. While this may have some impact in actual defect coverage at the physical level [19], switch-level stuck-at fault simulation of vectors created using these combinational models proved this modeling tradeoff still resulted in excellent stuck-at coverage while simplifying the remodeling effort.

While there was a heavy investment in automated remodeling of many types of cells using proprietary tools, a large amount of manual effort has still been necessary,

which has impacted the efficiency of the ATPG process. It is interesting to note that even though the amount of scan in the design was greatly increased compared to our past designs, the sheer variety and amount of custom circuits used in the design has also greatly increased the difficulty of remodeling effectively.

Memory Array Direct Access Testing

To aid in memory testing, direct access testing (DAT) [5] was used. The cache test hardware had to perform at greater than the design frequency, since it would be used to screen defects and characterize the arrays. The impact also had to be small with respect to die area and control wires. Therefore, existing data and control buses on the processor were reused for DAT. Additional control logic was added at each array for writes and reads while in DAT mode.

External bus pads were reused to control the DAT testing. Access was accomplished by using address pads and 64 data pads in the off-chip bus interface. The address pads are input only for DAT mode and are divided into groups that are used to select up to 2 array operations per bus clock. These operations are then mapped to core clocks based on a control register. The actions can be a no-operation, write, read and read without data return. Additional address pads are used to select the way and set in the array under test. This combination of actions on the array under test allows for testing all of the algorithms developed by Intel and HP to be performed. It also allows for great flexibility in addressing any future memory test algorithm that is developed as a result of debug and failure analysis work.

The 64 data pins are divided into two groups. One 32-bit group for input and the other 32-bit group for data output are used to pass data to/from the array under test. The input data is replicated as needed to provide a full cache line of data to the array under test. This replication is done internally in the chip and greatly reduces test time for larger arrays. Data returning from the array under test is compared to the 32-bit "master" chunk returned to the bus interface. The result from the internal comparison is stored in a "sticky" results register. The “sticky” register is read out at the end of the test to determine if any additional read bits failed. A 64-bit machine status register is used to select the array under test, the test mode, provides timing information and selects which 32 bit data chunk is returned to the external bus.

Test modes include direct access, diagnostic read/write access, array initialization, and built-in self-test (BIST). Direct access allows the replicated write and reads from the bus interface to and from the array under test. The entry under test is selected from the bus interface address pads.

Diagnostic mode allows for individual bit control in the array under test, which is useful for debugging while the processor is operating normally. A set of test registers is used to store the data to be written into or read from the array under test. The way and set to be used for the operation are stored in a control register. Writing the desired operation to a control register and executing an instruction to place the processor in a quiescent state ensures that the write/read operation does not conflict with normal processor operations. The control logic returns the processor to it normal operational state after the write/read operation has completed.

Initialization mode allows for the filling of the array under test with unique data or 1/0's. The built in self-test mode (BIST) is used to perform a March C- [20] algorithm on the L3 or L2 data arrays. BIST was chosen to provide toggle coverage on the arrays during burn-in and to provide a quick method of determining cache locations for redundancy replacement in the L3 array. The results of the BIST test are stored in the comparison registers used by the direct access test mode. The BIST mode is also used to test the L3 data array at power-on. This allows processor initialization code in a system to test the array before it is enabled. As the array testing is defined through registers accessible via scan, it can also be controlled via the IEEE 1149.1 test port, bus pads, assembly code or a combination to support use on a tester, in a system, or in the burn-in environment. Several arrays also implement a special weak write test mode feature, which speeds testing for memory retention faults as well as providing a more “active” test for defects in cells [21].

Results

The test circuitry described above was fully functional in first silicon. Scanpath shmoos show that parallel scanpath operations and the ITC/minitap logic work at frequencies up to 200 MHz (Figure 5). Individual access to scanpaths through the ITC was designed to only support operations at lower frequencies, and scanpath access in this mode works from below 1 MHz to 125 MHz. ATPG patterns were successfully run immediately on first silicon, including patterns for both static and dynamic circuit types.

DAT worked properly for all arrays and was a major benefit in providing excellent array coverage at first silicon as well as initialization capability for the processor. Boundary scan was also fully functional at first

silicon with the exception of the HIGHZ instruction, which failed to completely tristate the bus (this was quickly and easily fixed). No robustness issues have yet been discovered with the test circuitry implementation, and it appears that our validation efforts were successful.

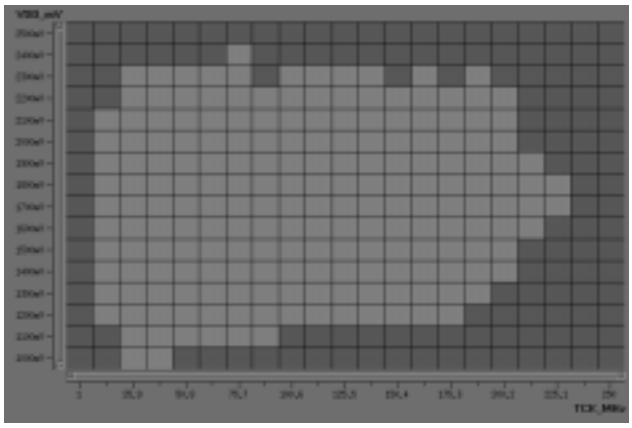

Figure 5 – Shmoo Plot of McKinley Scanpaths

Conclusion

The initial results of the McKinley testing process are very encouraging. All major test features worked in first silicon. The team was largely successful in achieving the goals set forth early in the methodology development. Design impact (both area and to the design team) was kept to a minimum, with negligible performance and area overhead (overhead of much less than 1% of the core circuit area for scan, the ITC and minitaps) and several fast and easy to run tools to verify the scan circuitry. Experiences in debugging the design indicate that sample on the fly and the full observability present in the design is very useful in finding root cause for design problems and marginalities rapidly (often within days).

The team was somewhat less successful in keeping the methodology consistent due to a plethora of custom latch designs. However, this was addressed when it happened during the physical design process with SPICE-based tools to ensure robust scan operation. Finally, while reasonable ATPG patterns at first silicon were available, the large number and variety of custom circuits present in the design have increased the difficulty of remodeling circuits properly for ATPG, even though ATPG should have been simplified through the large increase in scan capability. Efforts continue to improve stuck-at ATPG coverage through better modeling efforts and sequential ATPG, as well as experimenting with delay and bridging fault models. It is difficult to imagine being able to adequately test a design of this complexity rapidly without the use of extensive scan and dedicated test circuitry.

Acknowledgements

The McKinley processor is the result of a joint effort between Intel and Hewlett Packard engineers, and many dedicated and talented individuals were involved in the test methodology decisions and implementation. In particular, we would like to acknowledge the work of Dan Dixon, Sam Naffziger, Pavan Sunkerneni, Anurag Gupta, Rajgopal Narayanan, Joel Yuen, Jim Finnell and Kevin Laake in making the test methodology for the McKinley processor successful, as well as management support from Tom Meyer. Many other designers and test experts at Intel and HP also contributed to the design and implementation of the methodology. This work would not have been possible without them, and the results achieved are a testament to their expertise and skill.

References

[1] T. Ghani, et al., “100 nm Gate Length High Performance/Low Power CMOS Transistor Structure”, IEDM Technical Digest, 1999.

[2] J. Huck, et al., “Introducing the IA-64 Architecture”, IEEE Micro, September/October 2000, pp. 12-23.

[3] G. Hammond, S. Naffziger, “Next Generation Itanium Processor Overview”, Fall 2001 Intel Developer Forum, http://developer.intel.com

[4] D. Josephson, D. Dixon, B. Arnold, “Test Features of the PA7100LC Processor”, Proceedings of the International Test Conference, 1993, pp. 764-772.

[5] W. Needham, N. Gollakota, “DFT Strategy for Intel Microprocessors”, Proceedings of the International Test Conference, 1996, pp. 396-399.

[6] A. Carbine, D. Feltham, “Pentium® Pro Processor Design for Test and Debug”, Proceedings of the International Test Conference, 1997, pp. 294-303.

[7] J. Brauch, J. Fleischman, “Design of Cache Test Hardware on the HP PA8500”, Proceedings of the International Test Conference, 1997, pp. 286-293.

[8] M. Levitt, et al., “Testability, Debuggability, and Manufacturability Features of the UltraSPARC-I Microprocessor”, Proceedings of the International Test Conference, 1995, pp. 157-166.

[9] R. Fetherston, I. Shaik, S. Ma, “Testability Features of AMD-K6 Microprocessor”, Proceedings of the International Test Conference, 1997, pp. 408-413.

[10] L. Day, P. Ganfield, D. Rickert, F. Ziegler, “Test Methodology for a Microprocessor with Partial Scan”, Proceedings of the International Test Conference, 1998, pp. 708-716.

[11] M. Kusko, et al., “Microprocessor Test and Test Tool Methodology for the 500 MHz IBM S/390 G5 Chip”,

Proceedings of the International Test Conference, 1998, pp. 717-726.

[12] D. Bhavsar, J. Edmondson, “Testability Access of the High Speed Test Features in the Alpha 21264 Microprocessor”, Proceeding of the International Test Conference, 1998, pp. 487-495.

[13] T. Wood, “The Test and Debug Features of the AMD-K7 Microprocessor”, Proceedings of the International Test Conference, 1999, pp. 130-136.

[14] C. Pyron, et al., “DFT Advances in Motorola’s MPC7400, a PowerPC Microprocessor”, Proceedings of the International Test Conference, 1999, pp. 136-146.

[15] IEEE Standard 1149.1-1990 “IEEE Standard Test Access Port and Boundary-Scan Architecture”, IEEE Standards Board, New York, NY, 1990.

[16] D. Mukherjee, M. Pedram, M. Breuer, “Control Strategies for Chip-Based DFT/BIST Hardware”, Proceedings of the International Test Conference, 1994, pp. 893-895.

[17] A. Chandrakasan, W. Bowhill, F. Fox, “Design of High-Performance Microprocessor Circuits”, IEEE Press, 2001, ISBN 0-7803-6001-X.

[18] D. Harris, “Skew-Tolerant Circuit Design”, Morgan Kaufmann Publishers, 2001, ISBN 1-55860-636-X, pp. 55-57.

[19] R. Adams, E. Cooley, P. Hansen, “Quad DCVS Dynamic Logic Fault Modeling and Testing”, Proceedings of the International Test Conference, 1998, pp. 356-362.

[20] A. J. van de Goor, “Testing Semiconductor Memories”, John Wiley & Sons, Chichester, U.K., pp. 114-117.

[21] A. Meixner, J. Banik, “Weak Write Test Mode: An SRAM Cell Stability Design for Test Technique”, Proceedings of the International Test Conference, 1996, pp. 309-318.